Refactoring: Improving the Design of Existing Code

Martin Fowler

fowler@acm.org

www.martinfowler.com
www.thoughtworks.com

ThoughtWorks

What is Refactoring

A series of *small* steps, each of which changes the program's internal structure without changing its external behavior

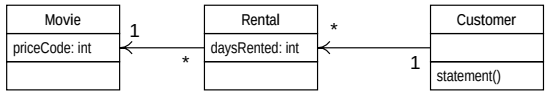
ThoughtWorks

Sample Output

```
Rental Record for Dinsdale Pirhana
  Monty Python and the Holy Grail   3.5
  Ran 2
  Star Trek 27        6
  Star Wars 3.2       3
  Wallace and Gromit  6
Amount owed is 20.5
You earned 6 frequent renter points
```

ThoughtWorks

Starting Class diagram



ThoughtWorks

Class Movie

```
public class Movie {
  public static final int  CHILDRENS = 2;
  public static final int  REGULAR = 0;
  public static final int  NEW_RELEASE = 1;

  private String _title;
  private int _priceCode;

  public Movie(String title, int priceCode) {
    _title = title;
    _priceCode = priceCode;
  }

  public int getPriceCode() {
    return _priceCode;
  }

  public void setPriceCode(int arg) {
    _priceCode = arg;
  }

  public String getTitle () {
    return _title;
  };
}
```

ThoughtWorks

Class Rental

```
class Rental {
    private Movie _movie;
    private int _daysRented;

    public Rental(Movie movie, int daysRented) {
        _movie = movie;
        _daysRented = daysRented;
    }
    public int getDaysRented() {
        return _daysRented;
    }
    public Movie getMovie() {
        return _movie;
    }
}
```

ThoughtWorks

Class Customer (almost)

```
class Customer {
    private String _name;
    private Vector _rentals = new Vector();

    public Customer (String name)  {
          _name = name;
    };

    public void addRental(Rental arg) {
                _rentals.addElement(arg);
    }
    public String getName ()       {
          return _name;
    };

    public String statement() // see next slide
```

ThoughtWorks

Customer.statement() part 1

```
public String statement() {
  double totalAmount = 0;
  int frequentRenterPoints = 0;
  Enumeration rentals = _rentals.elements();
  String result = "Rental Record for " + getName() + "\n";
  while (rentals.hasMoreElements()) {
    double thisAmount = 0;
    Rental each = (Rental) rentals.nextElement();

    //determine amounts for each line
    switch (each.getMovie().getPriceCode()) {
      case Movie.REGULAR:
        thisAmount += 2;
        if (each.getDaysRented() > 2)
          thisAmount += (each.getDaysRented() - 2) * 1.5;
        break;
      case Movie.NEW_RELEASE:
        thisAmount += each.getDaysRented() * 3;
        break;
      case Movie.CHILDRENS:
        thisAmount += 1.5;
        if (each.getDaysRented() > 3)
          thisAmount += (each.getDaysRented() - 3) * 1.5;
        break;

    }
```

continues on next slide

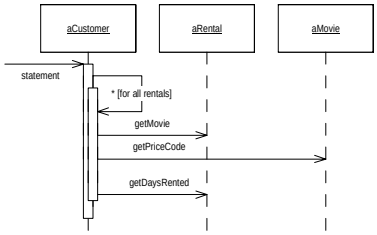
ThoughtWorks

Customer.statement() part 2

```
    // 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" +
    String.valueOf(thisAmount) + "\n";
    totalAmount += thisAmount;

  }
  //add footer lines
  result +=  "Amount owed is " + String.valueOf(totalAmount) + "\n";
  result += "You earned " + String.valueOf(frequentRenterPoints) +
        " frequent renter points";
  return result;

}
```

ThoughtWorks

Interactions for statement



ThoughtWorks

Requirements Changes

- **Produce an html version of the statement**
- **The movie classifications will soon change**
  - together with the rules for charging and for frequent renter points

ThoughtWorks

Extract Method

You have a code fragment that can be grouped together
Turn the fragment into a method whose name explains the purpose of the method.

```
void printOwing() {
    printBanner();

    //print details
    System.out.println ("name:" + _name);
    System.out.println ("amount" + getOutstanding());
}
```

⇓

```
void printOwing() {
    printBanner();
    printDetails(getOutstanding());
}

void printDetails (double outstanding) {
    System.out.println ("name:" + _name);
    System.out.println ("amount" + outstanding);
}
```

ThoughtWorks

Candidate Extraction

```
public String statement() {
  double totalAmount = 0;
  int frequentRenterPoints = 0;
  Enumeration rentals = _rentals.elements();
  String result = "Rental Record for " + getName() + "\n";
  while (rentals.hasMoreElements()) {
     double thisAmount = 0;
     Rental each = (Rental) rentals.nextElement();

     //determine amounts for each line
     switch (each.getMovie().getPriceCode()) {
       case Movie.REGULAR:
         thisAmount += 2;
         if (each.getDaysRented() > 2)
           thisAmount += (each.getDaysRented() - 2) * 1.5;
         break;
       case Movie.NEW_RELEASE:
         thisAmount += each.getDaysRented() * 3;
         break;
       case Movie.CHILDRENS:
         thisAmount += 1.5;
         if (each.getDaysRented() > 3)
           thisAmount += (each.getDaysRented() - 3) * 1.5;
         break;

     }

           [snip]12
```


Steps for *Extract Method*

- **Create method named after intention of code**
- **Copy extracted code**
- **Look for local variables and parameters**
  - turn into parameter
  - turn into return value
  - declare within method
- **Compile**
- **Replace code fragment with call to new method**
- **Compile and test**

Extracting the Amount Calculation

```
private int amountFor(Rental each) {
    int thisAmount = 0;
    switch (each.getMovie().getPriceCode()) {
        case Movie.REGULAR:
            thisAmount += 2;
            if (each.getDaysRented() > 2)
                thisAmount += (each.getDaysRented() - 2) *
                        1.5;
            break;
        case Movie.NEW_RELEASE:
            thisAmount += each.getDaysRented() * 3;
            break;
        case Movie.CHILDRENS:
            thisAmount += 1.5;
            if (each.getDaysRented() > 3)
                thisAmount += (each.getDaysRented() - 3) *
                        1.5;
            break;
    }
    return thisAmount;
}
```


Statement() after extraction

```
public String statement() {
    double totalAmount = 0;
    int frequentRenterPoints = 0;
    Enumeration rentals = _rentals.elements();
    String result = "Rental Record for " + getName() + "\n";
    while (rentals.hasMoreElements()) {
        double thisAmount = 0;
        Rental each = (Rental) rentals.nextElement();

        thisAmount = amountFor(each);

        // 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" +
            String.valueOf(thisAmount) + "\n";
        totalAmount += thisAmount;

    }
    //add footer lines
    result +=  "Amount owed is " + String.valueOf(totalAmount) + "\n";
    result += "You earned " + String.valueOf(frequentRenterPoints) +
        " frequent renter points";
    return result;

 }
}
```


Extracting the amount calculation (2)

```
private double amountFor(Rental each) {
    double thisAmount = 0;
    switch (each.getMovie().getPriceCode()) {
        case Movie.REGULAR:
            thisAmount += 2;
            if (each.getDaysRented() > 2)
                thisAmount += (each.getDaysRented() - 2) * 1.5;
            break;
        case Movie.NEW_RELEASE:
            thisAmount += each.getDaysRented() * 3;
            break;
        case Movie.CHILDRENS:
            thisAmount += 1.5;
            if (each.getDaysRented() > 3)
                thisAmount += (each.getDaysRented() - 3) * 1.5;
            break;
    }
    return thisAmount;
}
```


Change names of variables

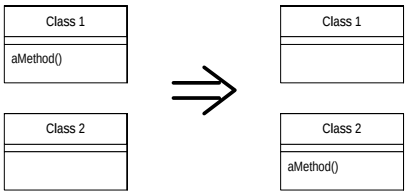
```
private double amountFor(Rental aRental) {
    double result = 0;
    switch (aRental.getMovie().getPriceCode()) {
        case Movie.REGULAR:
            result += 2;
            if (aRental.getDaysRented() > 2)
                result += (aRental.getDaysRented() - 2) * 1.5;
            break;
        case Movie.NEW_RELEASE:
            result += aRental.getDaysRented() * 3;
            break;
        case Movie.CHILDRENS:
            result += 1.5;
            if (aRental.getDaysRented() > 3)
                result += (aRental.getDaysRented() - 3) * 1.5;
            break;
    }
    return result;
}
```

Is this important?

Is this method in the right place?

Move Method

A method is, or will be, using or used by more features of another class than the class it is defined on.
Create a new method with a similar body in the class it uses most. Either turn the old method into a simple delegation, or remove it altogether.



ThoughtWorks

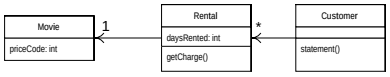
Steps for *Move method*

- **Declare method in target class**
- **Copy and fit code**
- **Set up a reference from the source object to the target**
- **Turn the original method into a delegating method**
  - `amountOf(Rental each) {return each.charge()}`
  - Check for overriding methods
- **Compile and test**
- **Find all users of the method**
  - Adjust them to call method on target
- **Remove original method**
- **Compile and test**

ThoughtWorks

Moving amount() to Rental

```
class Rental                    ...
  double getCharge() {
      double result = 0;
      switch (getMovie().getPriceCode()) {
          case Movie.REGULAR:
              result += 2;
              if (getDaysRented() > 2)
                  result += (getDaysRented() - 2) * 1.5;
              break;
          case Movie.NEW_RELEASE:
              result += getDaysRented() * 3;
              break;
          case Movie.CHILDRENS:
              result += 1.5;
              if (getDaysRented() > 3)
                  result += (getDaysRented() - 3) * 1.5;
              break;
      }
      return result;
  }
```



ThoughtWorks

Altered statement

```
class Customer...
    public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";
        while (rentals.hasMoreElements()) {
            double thisAmount = 0;
            Rental each = (Rental) rentals.nextElement();

          thisAmount = each.getCharge();
            // 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" +
          String.valueOf(thisAmount) + "\n";
            totalAmount += thisAmount;

        }
        //add footer lines
        result +=  "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) +
        " frequent renter points";
        return result;

    }
```

ThoughtWorks

Problems with temps

```
class Customer...
    public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";
        while (rentals.hasMoreElements()) {
            double thisAmount = 0;
            Rental each = (Rental) rentals.nextElement();

          thisAmount = each.getCharge();
            // 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" +
          String.valueOf(thisAmount) + "\n";
            totalAmount += thisAmount;

        }
        //add footer lines
        result +=  "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) +
        " frequent renter points";
        return result;

    }
```

ThoughtWorks

Replace Temp with Query

You are using a temporary variable to hold the result of an expression.
Extract the expression into a method. Replace all references to the temp with the expression. The new method can then be used in other methods.

```
        double basePrice = _quantity * _itemPrice;
        if (basePrice > 1000)
            return basePrice * 0.95;
        else
            return basePrice * 0.98;
```

```
        if (basePrice() > 1000)
            return basePrice() * 0.95;
        else
            return basePrice() * 0.98;
...
    double basePrice() {
        return _quantity * _itemPrice;
    }
```

ThoughtWorks

Steps for *Replace temp with Query*

- Find temp with a single assignment
- Extract Right Hand Side of assignment
- Replace all references of temp with new method
- Remove declaration and assignment of temp
- Compile and test

ThoughtWorks

thisAmount removed

```
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";
        while (rentals.hasMoreElements()) {
            Rental each = (Rental) rentals.nextElement();

            // 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" +
               String.valueOf(each.getCharge()) + "\n";
            totalAmount += each.getCharge();

        }
        //add footer lines
        result +=  "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) +
            " frequent renter points";
        return result;

    }
}
```

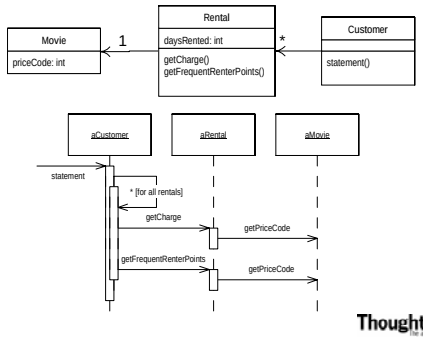
ThoughtWorks

Extract and move frequentRenterPoints()

```
class Customer...
    public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";
        while (rentals.hasMoreElements()) {
            Rental each = (Rental) rentals.nextElement();
            frequentRenterPoints += each.getFrequentRenterPoints();

            //show figures for this rental
            result += "\t" + each.getMovie().getTitle()+ "\t" +
                    String.valueOf(each.getCharge()) + "\n";
            totalAmount += each.getCharge();
        }

        //add footer lines
        result +=  "Amount owed is " + String.valueOf(totalAmount) +
"\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) +
                " frequent renter points";
        return result;
    }
```

ThoughtWorks

After moving charge and frequent renter points



ThoughtWorks

More temps to kill

```
class Customer...
    public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";
        while (rentals.hasMoreElements()) {
            Rental each = (Rental) rentals.nextElement();
            frequentRenterPoints += each.getFrequentRenterPoints();

            //show figures for this rental
            result += "\t" + each.getMovie().getTitle()+ "\t" +
                    String.valueOf(each.getCharge()) + "\n";
            totalAmount += each.getCharge();
        }

        //add footer lines
        result +=  "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned " + String.valueOf(frequentRenterPoints) +
                " frequent renter points";
        return result;
    }
```

ThoughtWorks

The new methods

```
class Customer…

 private double getTotalCharge() {
        double result = 0;
        Enumeration rentals = _rentals.elements();
        while (rentals.hasMoreElements()) {
            Rental each = (Rental) rentals.nextElement();
            result += each.getCharge();
        }
        return result;
 }

 private int getTotalFrequentRenterPoints(){
        int result = 0;
        Enumeration rentals = _rentals.elements();
        while (rentals.hasMoreElements()) {
            Rental each = (Rental) rentals.nextElement();
            result += each.getFrequentRenterPoints();
        }
        return result;
 }
```

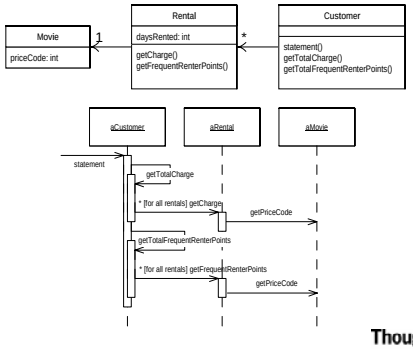
ThoughtWorks

The temps removed

```
public String statement() {
    Enumeration rentals = _rentals.elements();
    String result = "Rental Record for " + getName() + "\n";
    while (rentals.hasMoreElements()) {
       Rental each = (Rental) rentals.nextElement();


       //show figures for this rental
       result += "\t" + each.getMovie().getTitle()+ "\t" +
          String.valueOf(each.getCharge()) + "\n";
       }

    //add footer lines
    result +=  "Amount owed is " + String.valueOf(getTotalCharge()) + "\n";
    result += "You earned " + String.valueOf(getTotalFrequentRenterPoints())
+
       " frequent renter points";
    return result;
   }
```

ThoughtWorks

After replacing the totals



ThoughtWorks

htmlStatement()

```
public String htmlStatement() {
   Enumeration rentals = _rentals.elements();
   String result = "<H1>Rentals for <EM>" + getName() + "</EM></H1><P>\n";
   while (rentals.hasMoreElements()) {
      Rental each = (Rental) rentals.nextElement();
      //show figures for each rental
      result += each.getMovie().getTitle()+ ": " +
                  String.valueOf(each.getCharge()) + "<BR>\n";
   }
   //add footer lines
   result +=  "<P>You owe <EM>" + String.valueOf(getTotalCharge()) +
                           "</EM><P>\n";
   result += "On this rental you earned <EM>" +
      String.valueOf(getTotalFrequentRenterPoints()) +
      "</EM> frequent renter points<P>";
   return result;
 }
```

ThoughtWorks

The current getCharge method

```
class Rental         ...
  double getCharge() {
      double result = 0;
      switch (getMovie().getPriceCode()) {
          case Movie.REGULAR:
              result += 2;
              if (getDaysRented() > 2)
                  result += (getDaysRented() - 2) * 1.5;
              break;
          case Movie.NEW_RELEASE:
              result += getDaysRented() * 3;
              break;
          case Movie.CHILDRENS:
              result += 1.5;
              if (getDaysRented() > 3)
                  result += (getDaysRented() - 3) * 1.5;
              break;
      }
      return result;
  }
```

ThoughtWorks

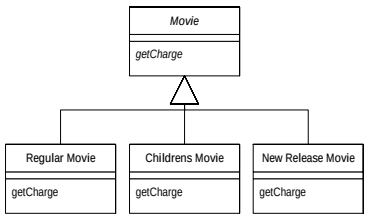
getCharge moved to Movie

```
class Rental...
   double getCharge() {
      return _movie.getCharge(_daysRented);
   }


class Movie …
   double getCharge(int daysRented) {
       double result = 0;
       switch (getPriceCode()) {
           case Movie.REGULAR:
               result += 2;
               if (daysRented > 2)
                   result += (daysRented - 2) * 1.5;
               break;
           case Movie.NEW_RELEASE:
               result += daysRented * 3;
               break;
           case Movie.CHILDRENS:
               result += 1.5;
               if (daysRented > 3)
                   result += (daysRented - 3) * 1.5;
               break;
       }
       return result;
   }
```

Do the same with frequentRenterPoints()

ThoughtWorks

Consider inheritance



How's this?

ThoughtWorks

The Two Hats

Adding Function

- Add new capabilities to the system
- Adds new tests
- Get the test working

Refactoring

- Does not add any new features
- Does not add tests (but may change some)
- Restructure the code to remove redundancy

Swap frequently between the hats, but only wear one at a time

ThoughtWorks

Why Refactor

- **To improve the software design**
  - combat's "bit rot"
  - makes the program easier to change
- **To make the software easier to understand**
  - write for people, not the compiler
  - understand unfamiliar code
- **To help find bugs**
  - refactor while debugging to clarify the code

Refactoring helps you program faster!

ThoughtWorks

When should you refactor?

- **To add new functionality**
  - refactor existing code until you understand it
  - refactor the design to make it easy to add
- **To find bugs**
  - refactor to understand the code
- **For code reviews**
  - immediate effect of code review
  - allows for higher level suggestions

Don't set aside time for refactoring, include it in your normal activities

ThoughtWorks