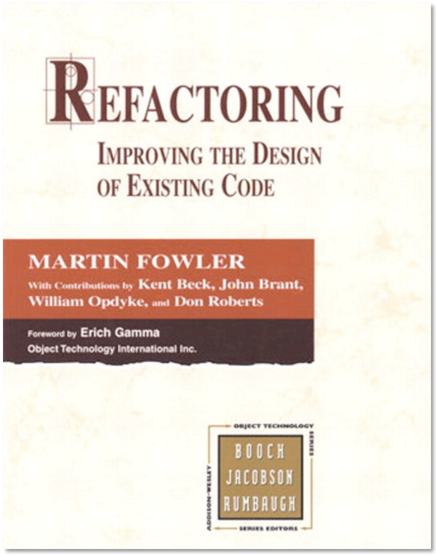
Refactoring

Baris Aktemur

CS 534 | Ozyegin University

Contents from "Refactoring: Improving the Design of Existing Code" by Martin Fowler


REFACTORING
IMPROVING THE DESIGN OF EXISTING CODE
MARTIN FOWLER
With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International Inc.
OBJECT TECHNOLOGY SERIES
BOOCH JACOBSON RUMBAUGH
SERIES EDITORS
ADDISON-WESLEY

Definition

- A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.

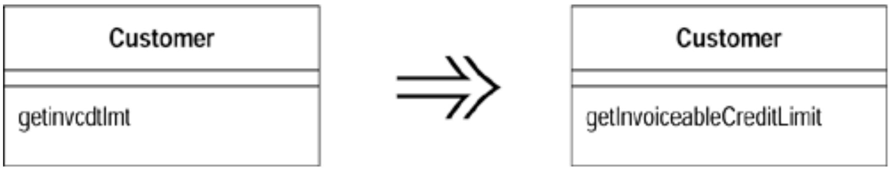
- When you find you have to add a feature to a program, and the program's code is not structured in a convenient way to add the feature, first refactor the program to make it easy to add the feature, then add the feature.
- Before you start refactoring, check that you have a solid suite of tests. These tests must be self-checking.

- Anyone can write code that a computer can understand. Only good programmers write code that humans can understand.

Code Smells

- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Switch Statements
- Parallel Inheritance Hierarchies
- Message Chains
- Middle Man
- etc...

Rename Method



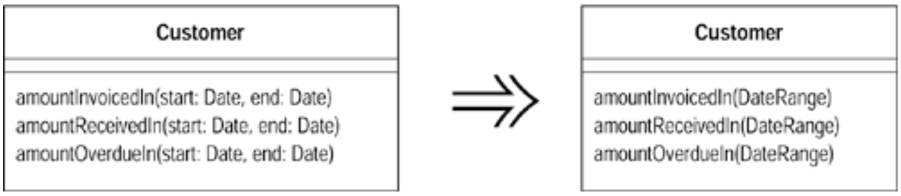
Customer

getinvcdtlmt

Customer

getInvoiceableCreditLimit

Introduce Parameter Object



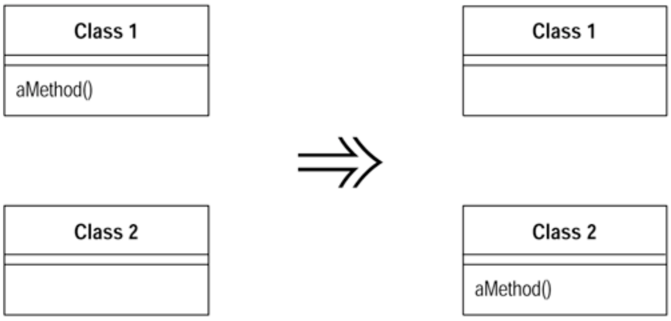
Customer

amountInvoicedIn(start: Date, end: Date)
amountReceivedIn(start: Date, end: Date)
amountOverdueIn(start: Date, end: Date)

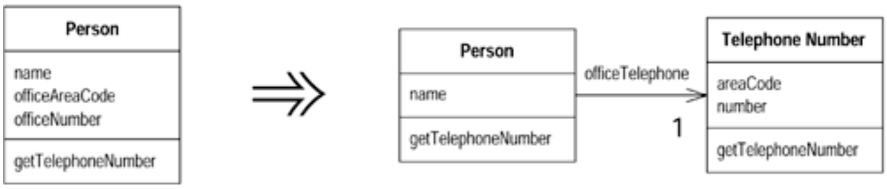
Customer

amountInvoicedIn(DateRange)
amountReceivedIn(DateRange)
amountOverdueIn(DateRange)

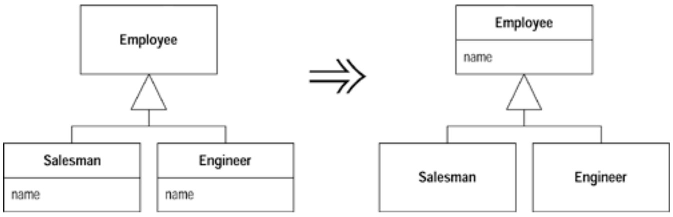
Move Method



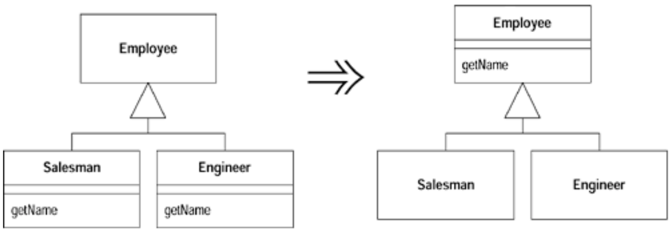
Extract Class

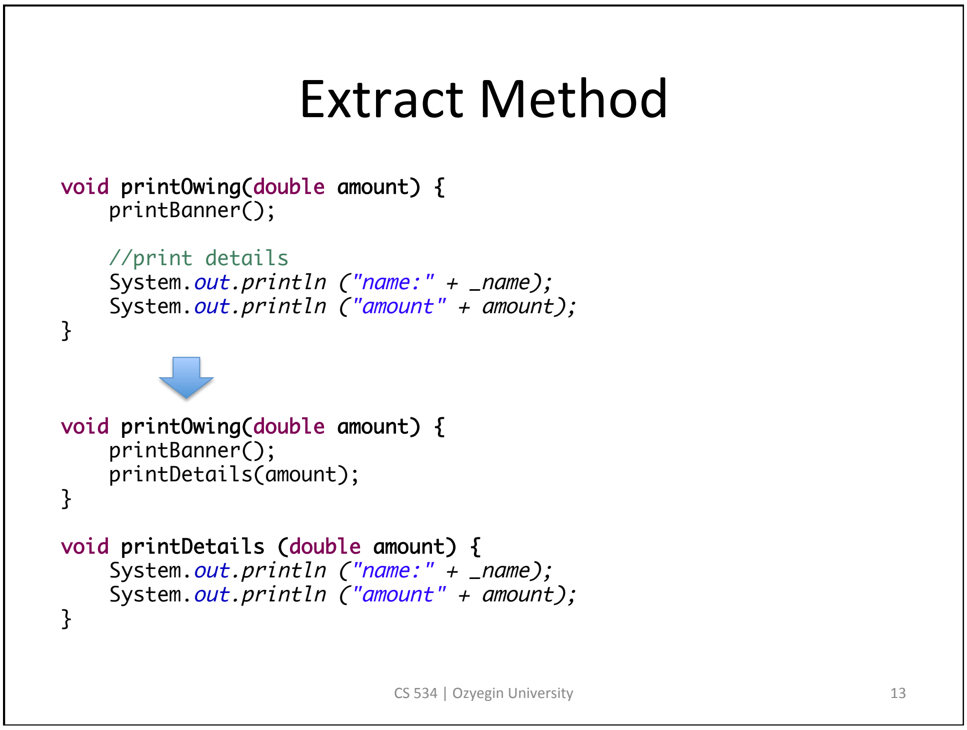


Pull Up Field



Pull Up Method

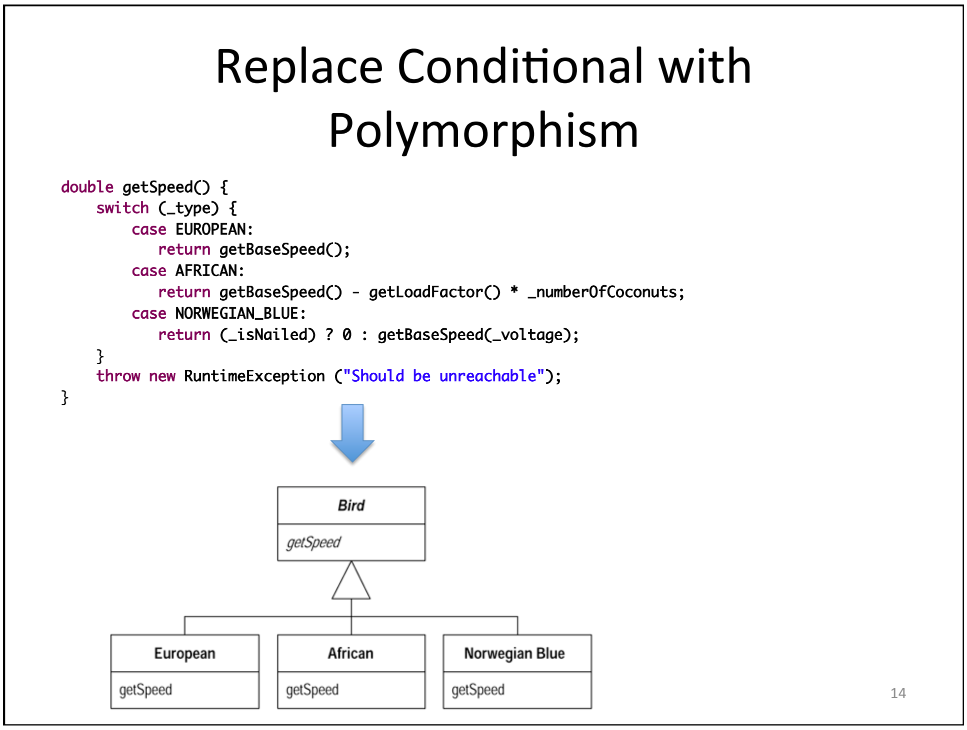




Extract Method

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

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

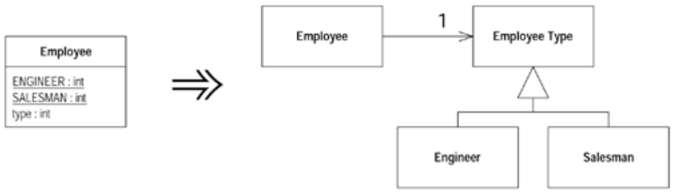
```
void printOwing(double amount) {
    printBanner();
    printDetails(amount);
}

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



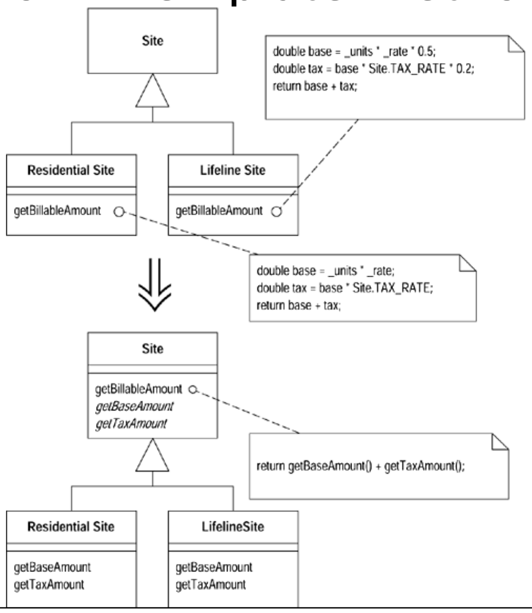
Replace Conditional with Polymorphism

```
double getSpeed() {
    switch (_type) {
        case EUROPEAN:
           return getBaseSpeed();
        case AFRICAN:
           return getBaseSpeed() - getLoadFactor() * _numberOfCoconuts;
        case NORWEGIAN_BLUE:
           return (_isNailed) ? 0 : getBaseSpeed(_voltage);
    }
    throw new RuntimeException ("Should be unreachable");
}
```

Replace Type Code with State/Strategy



Form Template Method



```
double base = _units * _rate * 0.5;
double tax = base * Site.TAX_RATE * 0.2;
return base + tax;
```

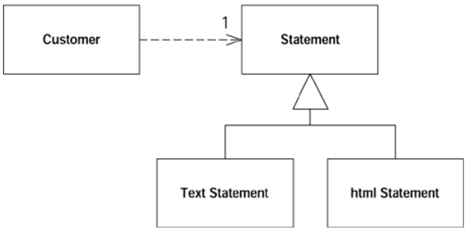
```
double base = _units * _rate;
double tax = base * Site.TAX_RATE;
return base + tax;
```

```
return getBaseAmount() + getTaxAmount();
```

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

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()+ ": " + each.getCharge() + "<BR>\n";
    }
    //add footer lines
    result +=  "<P>You owe <EM>” + getTotalCharge() + "</EM><P>\n";
    result += "On this rental you earned <EM>" + getTotalFrequentRenterPoints() +
        "</EM> frequent renter points<P>";
    return result;
}
```

Use Strategy…

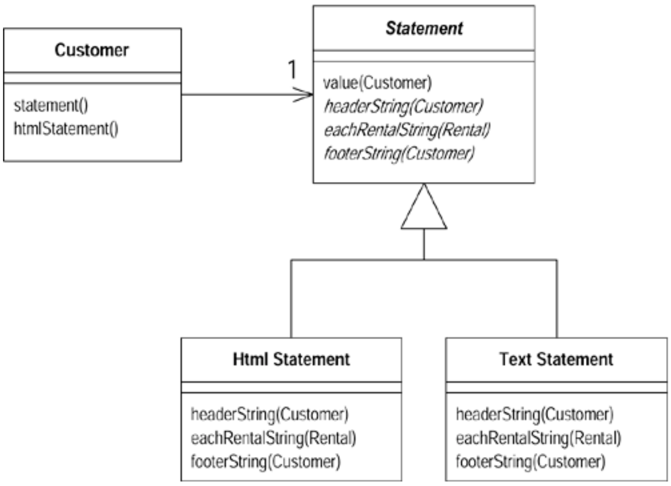
```
class Statement {}
class TextStatement extends Statement {}
class HtmlStatement extends Statement {}

class Customer...
public String statement() {
  return new TextStatement().value(this);
}
public String htmlStatement() {
  return new HtmlStatement().value(this);
}
```



```
class HtmlStatement…
public String value(Customer aCustomer) {
      Enumeration rentals = aCustomer.getRentals();
      String result = headerString(aCustomer);
      while (rentals.hasMoreElements()) {
          Rental each = (Rental) rentals.nextElement();
          result += eachRentalString(each);
      }
      result +=  footerString(aCustomer);
      return result;
}
String eachRentalString (Rental aRental) {
    return aRental.getMovie().getTitle()+ ": " +
        String.valueOf(aRental.getCharge()) + "<BR>\n";
}
String footerString (Customer aCustomer) {
      return "<P>You owe <EM>" +
     String.valueOf(aCustomer.getTotalCharge()) +
      "</EM><P>
" + "On this rental you earned <EM>" +

     String.valueOf(aCustomer.getTotalFrequentRenterPoin
     ts()) +
        "</EM> frequent renter points<P>";
}
```

```
class TextStatement …
public String value(Customer aCustomer) {
    Enumeration rentals = aCustomer.getRentals();
    String result = headerString(aCustomer);
    while (rentals.hasMoreElements()) {
        Rental each = (Rental) rentals.nextElement();
        result += eachRentalString(each);
    }
    result +=  footerString(aCustomer);
    return result;
 }
String eachRentalString (Rental aRental) {
    return "\t" + aRental.getMovie().getTitle()+ "\t" +
        String.valueOf(aRental.getCharge()) + "\n";
  }
String footerString (Customer aCustomer) {
      return "Amount owed is " +
String.valueOf(aCustomer.getTotalCharge()) + "\n" +
         "You earned " +
String.valueOf(aCustomer.getTotalFrequentRenterPoints())
+
         " frequent renter points";
  }
```

Pull Up Common Method



Customer

statement()
htmlStatement()

1

Statement

value(Customer)
headerString(Customer)
eachRentalString(Rental)
footerString(Customer)

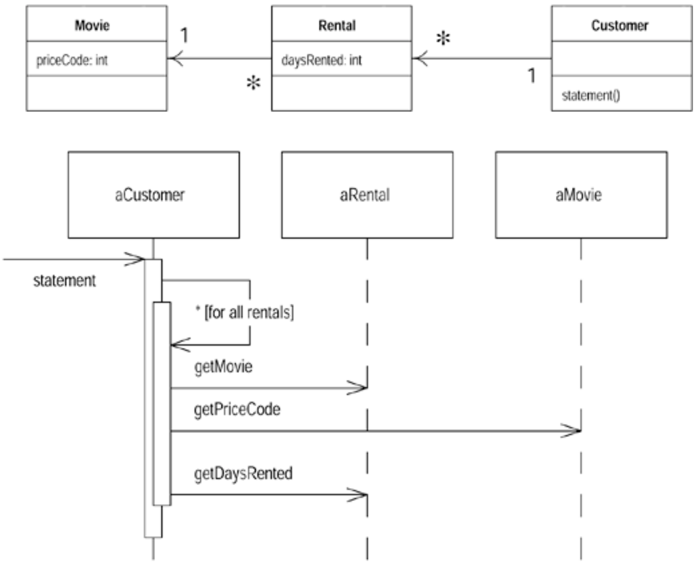
Html Statement

headerString(Customer)
eachRentalString(Rental)
footerString(Customer)

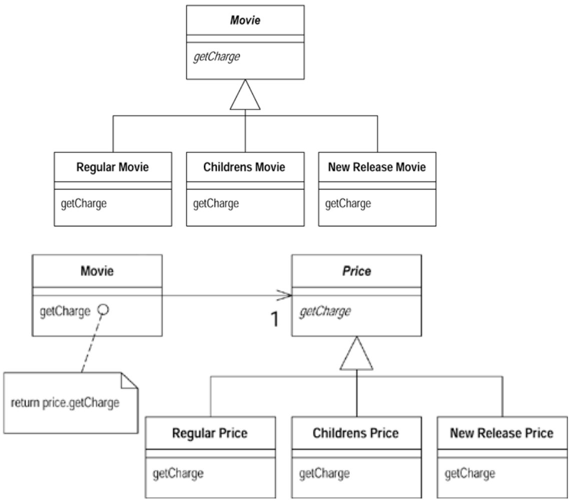
Text Statement

headerString(Customer)
eachRentalString(Rental)
footerString(Customer)

Case Study

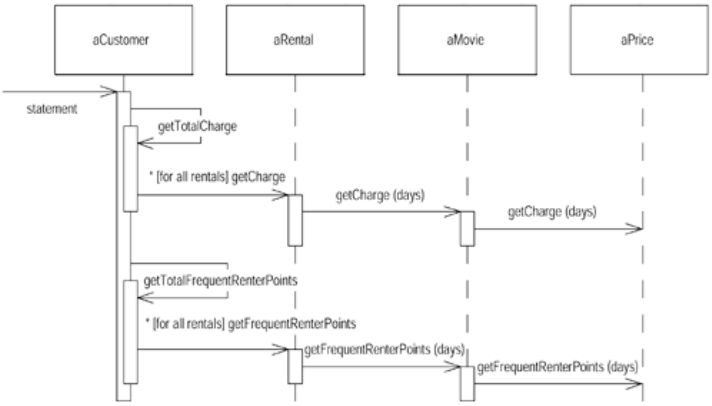


After several refactorings...

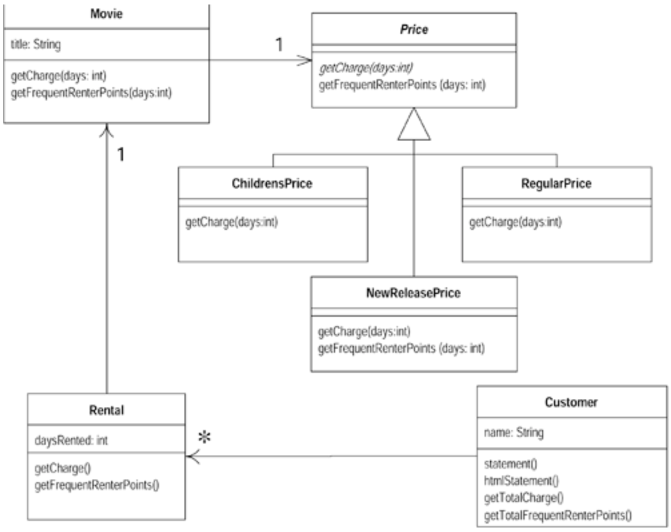


Which one?

After the State Pattern


aCustomer
aRental
aMovie
aPrice
statement
getTotalCharge
* [for all rentals] getCharge
getCharge (days)
getCharge (days)
getTotalFrequentRenterPoints
* [for all rentals] getFrequentRenterPoints
getFrequentRenterPoints (days)
getFrequentRenterPoints (days)

After the State Pattern


Movie
title: String
getCharge(days: int)
getFrequentRenterPoints(days:int)
1
Price
getCharge(days:int)
getFrequentRenterPoints (days: int)
1
ChildrensPrice
getCharge(days:int)
RegularPrice
getCharge(days:int)
NewReleasePrice
getCharge(days:int)
getFrequentRenterPoints (days: int)
Rental
daysRented: int
getCharge()
getFrequentRenterPoints()
*
Customer
name: String
statement()
htmlStatement()
getTotalCharge()
getTotalFrequentRenterPoints()