Refactoring: Improving the Design of Existing Code

Martin Fowler

fowler@acm.org

www.martinfowler.com
www.thoughtworks.com

ThoughtWorks®
The art of heavy lifting.™

What is Refactoring

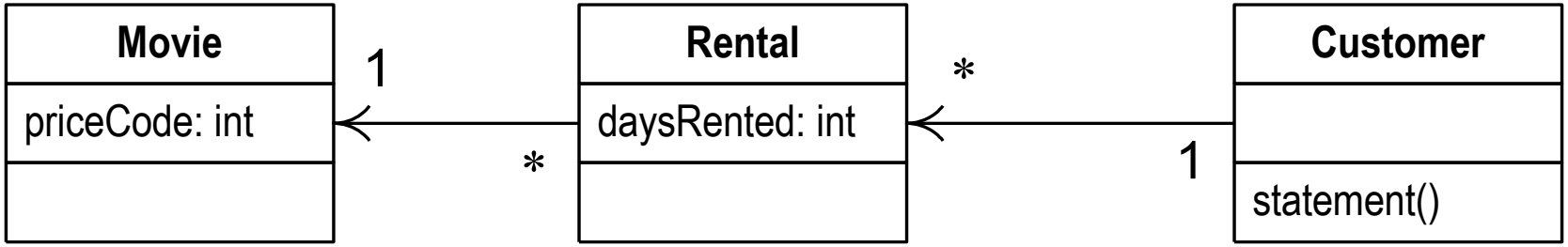
A series of *small* steps, each of which changes the program's internal structure without changing its external behavior

- **Verify no change in external behavior by**
  - testing
  - formal code analysis by tool
- **In practice good tests are essential**

Sample Output

```
Rental Record for Dinsdale Pirhana
  Monty Python and the Holy Grail   3.5
  Star Trek 27          6
  Star Wars 3.2         3
  Wallace and Gromit  6
Amount owed is 18.5
You earned 5 frequent renter points
```

Starting Class diagram



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;
  };
}
```

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;
    }
}
```

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
```

Customer.statement() part 1

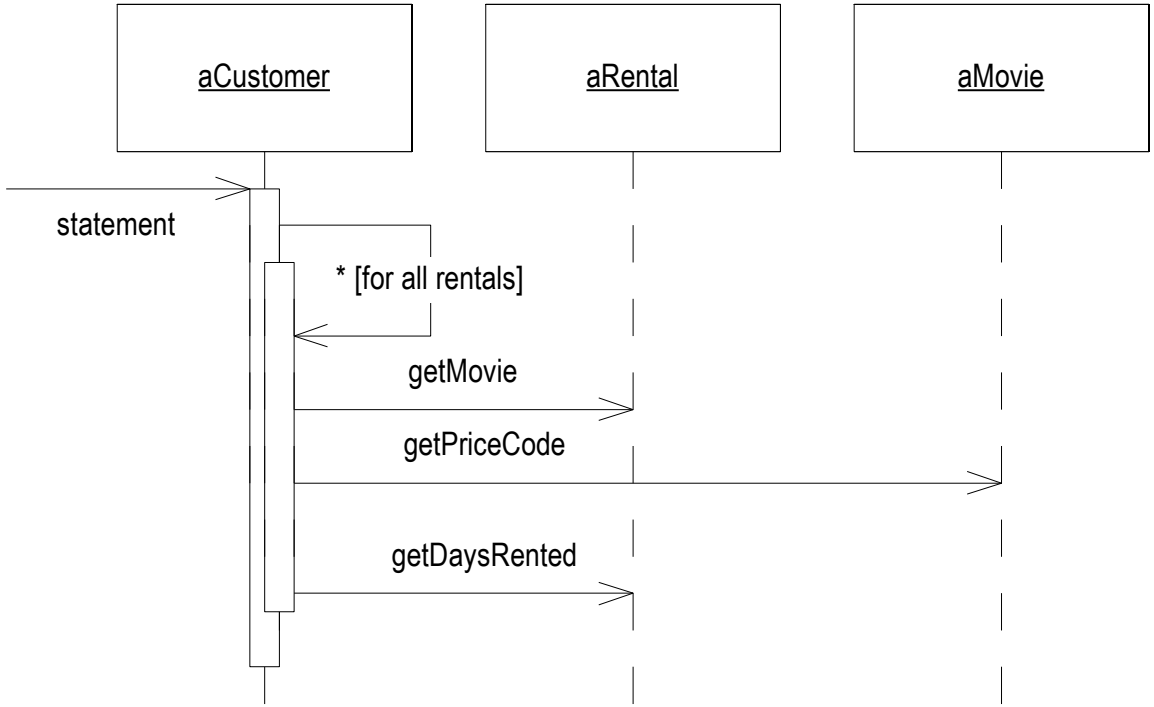
```
public String statement() {
   double totalAmount = 0;
   int frequentRenterPoints = 0;
   Enumeration rentals = _rentals.elements();
   String result = "Rental Record for " + getName() + "\n";
   while (rentals.hasMoreElements()) {
      double thisAmount = 0;
      Rental each = (Rental) rentals.nextElement();

      //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

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;

}
```

Interactions for statement



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

You need tests

- **Use a simple test framework to write and organize tests**
  - http://junit.org
  - http://xprogramming.com/software
- **Small fast tests for code you're working on**
- **Complete tests for build**
  - Run full test suite as part of build process
  - http://martinfowler.com/articles/continuousIntegration.html
- **Build tests as you go for legacy code**

Candidate Extraction

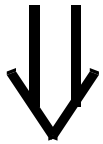
```
public String statement() {
   double totalAmount = 0;
   int frequentRenterPoints = 0;
   Enumeration rentals = _rentals.elements();
   String result = "Rental Record for " + getName() + "\n";
   while (rentals.hasMoreElements()) {
      double thisAmount = 0;
      Rental each = (Rental) rentals.nextElement();

      //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
```

ThoughtWorks®
The art of heavy lifting.™

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);
}
```

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;

 }
}
```

ThoughtWorks®
The art of heavy lifting.™

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?

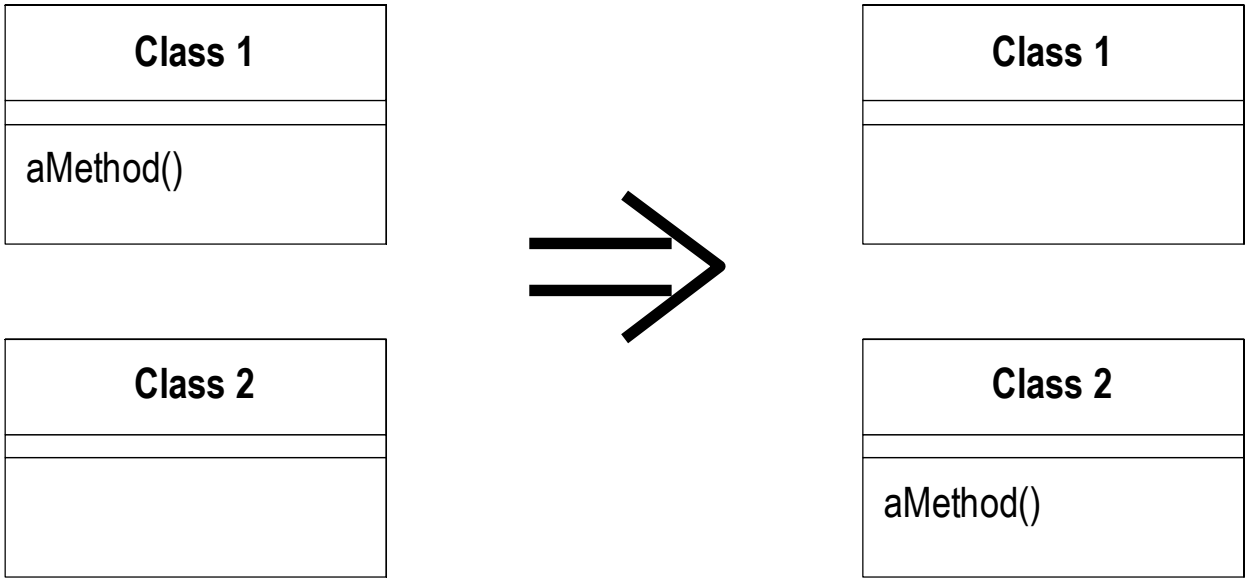
Is this method in the right place?

ThoughtWorks®
The art of heavy lifting.™

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.

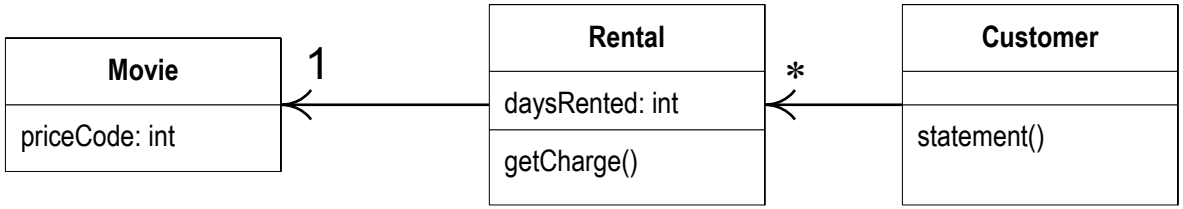


Steps for *Move method*

- **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**

Moving amountFor() 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;
   }
```



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®
The art of heavy lifting.™

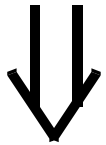
Problems with temps

```
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;
    }
```

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®
The art of heavy lifting.™

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**

thisAmount removed

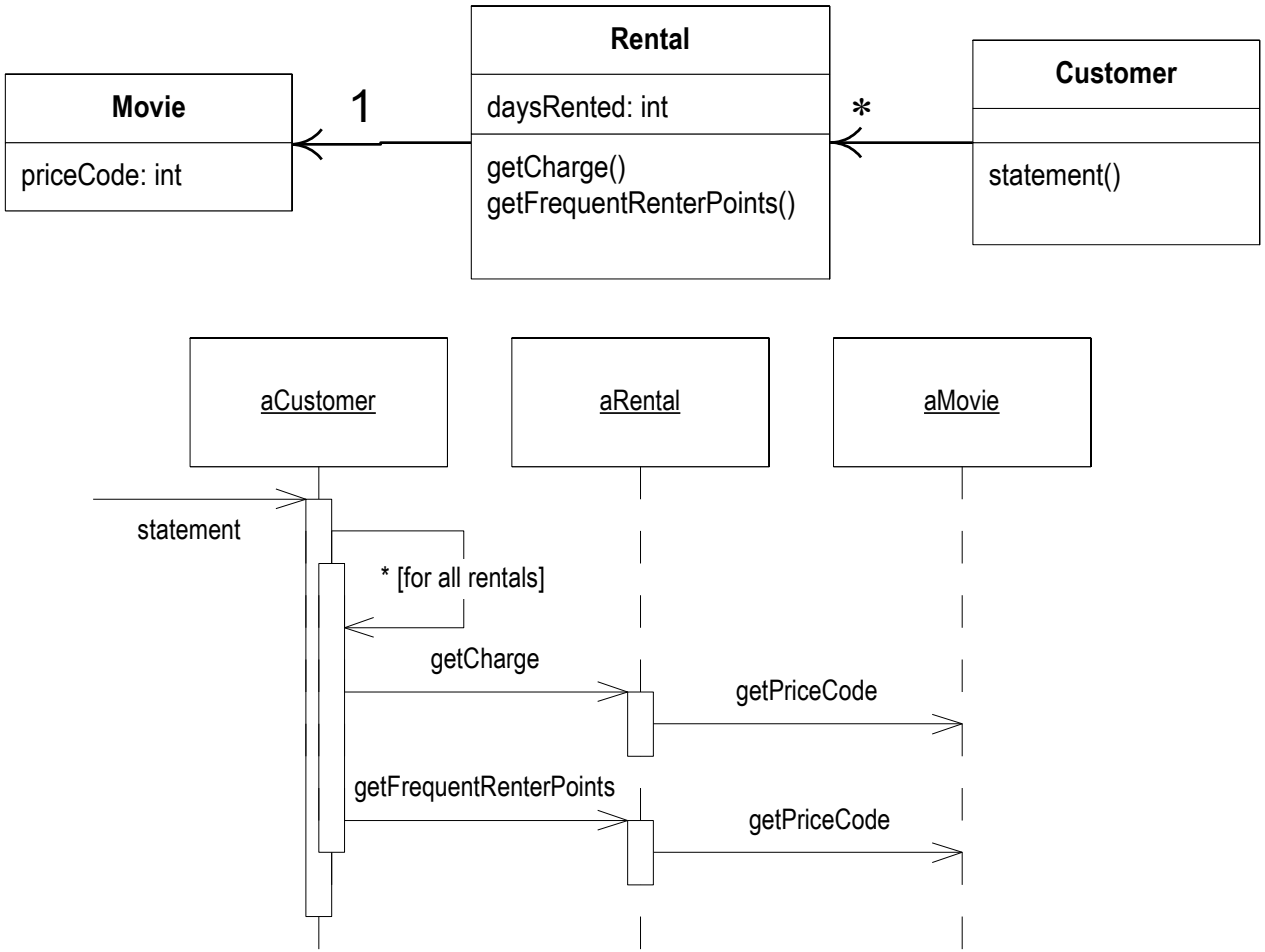
```
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        Enumeration rentals = _rentals.elements();
        String result = "Rental Record for " + getName() + "\n";
        while (rentals.hasMoreElements()) {
            Rental each = (Rental) rentals.nextElement();

            // 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®
The art of heavy lifting.™

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;
    }
```

After moving charge and frequent renter points



ThoughtWorks
The art of heavy lifting.

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 art of heavy lifting.™

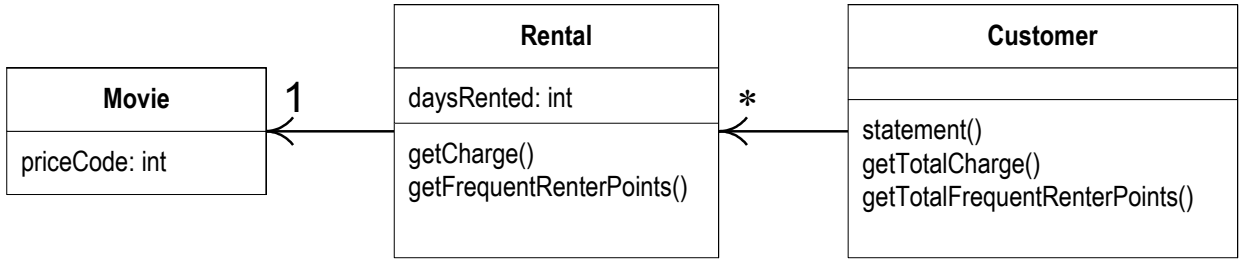
The new methods

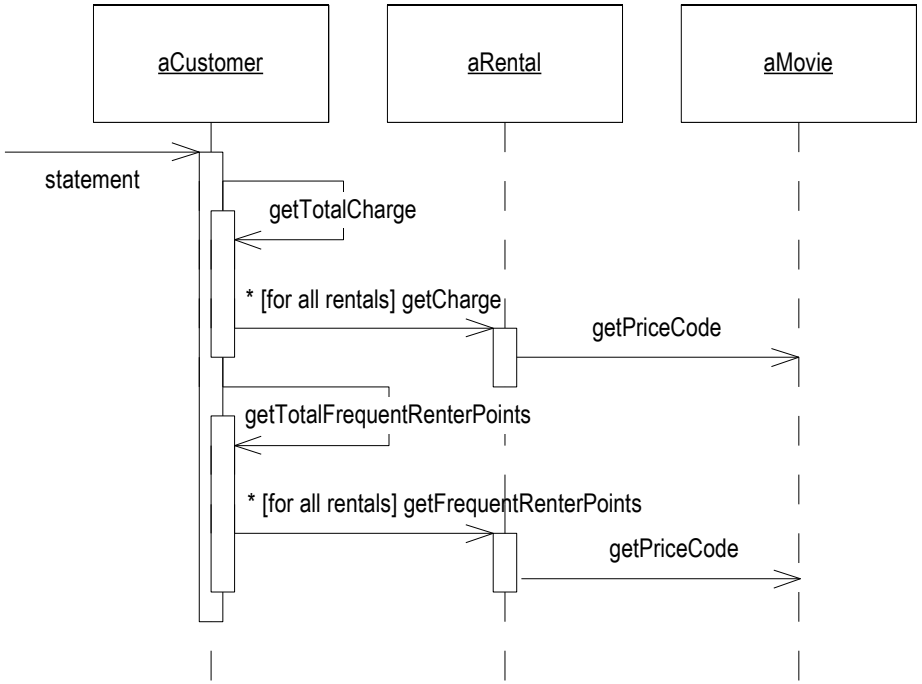
```
class Customer…

 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;
  }
```

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;
}
```

After replacing the totals





ThoughtWorks®
The art of heavy lifting.™

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;
}
```

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;
   }
```

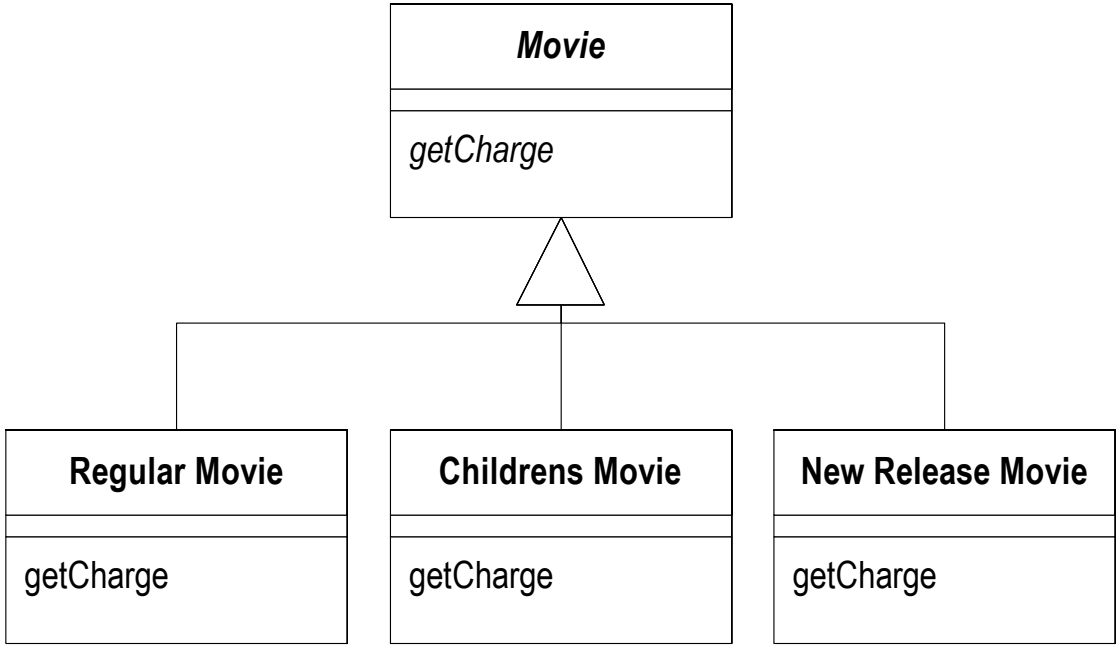
getCharge moved to Movie

```
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 getFrequentRenterPoints()

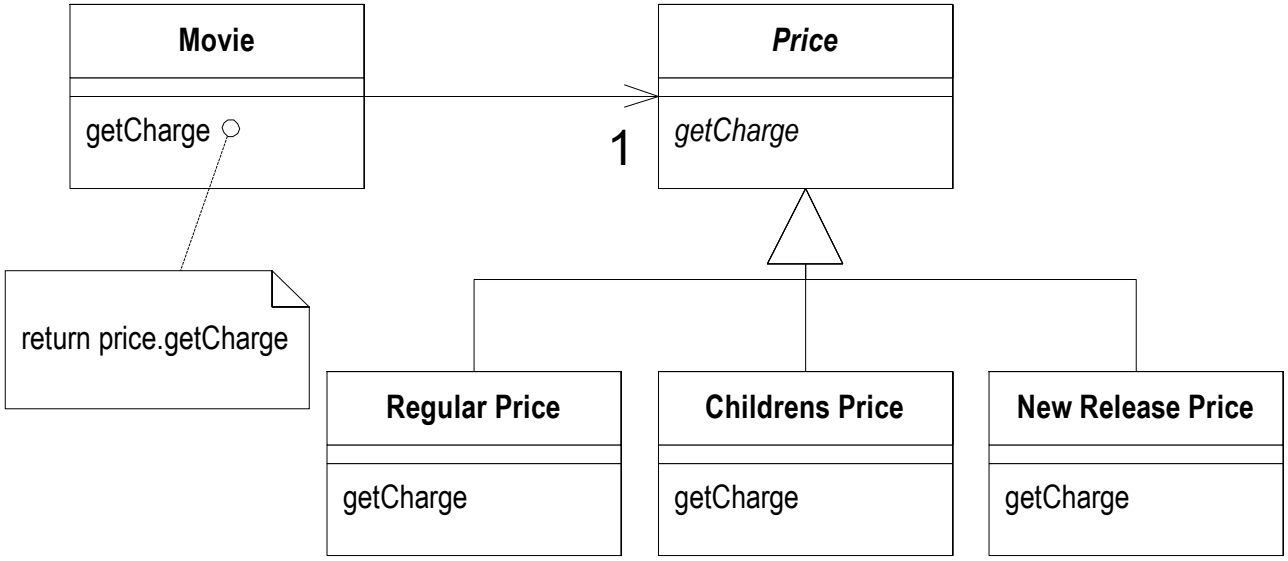
ThoughtWorks®
The art of heavy lifting.™

Consider inheritance



How's this?

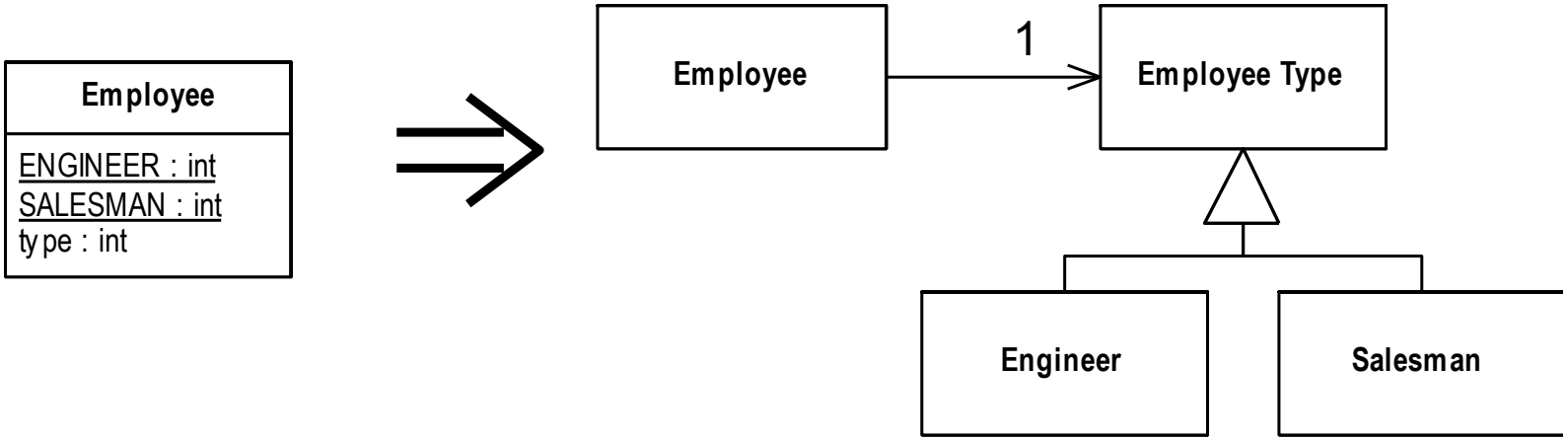
Using the State Pattern



Replace Type Code with State/Strategy

You have a type code which affects the behavior of a class but you cannot use subclassing.

Replace the type code with a state object.



Price codes on the price hierarchy

```
abstract class Price {
   abstract int getPriceCode();
}
class ChildrensPrice extends Price {
   int getPriceCode() {
      return Movie.CHILDRENS;
   }
}
class NewReleasePrice extends Price {
   int getPriceCode() {
      return Movie.NEW_RELEASE;
   }
}
class RegularPrice extends Price {
   int getPriceCode() {
      return Movie.REGULAR;
   }
}
```

Change accessors on Movie

```
public int getPriceCode() {
   return _priceCode;
}
public setPriceCode (int arg) {
   _priceCode = arg;
}
private int _priceCode;
```

```
public int getPriceCode() {
   return _price.getPriceCode();
}
public void setPriceCode(int arg) {
   switch (arg) {
      case REGULAR:
         _price = new RegularPrice();
         break;
      case CHILDRENS:
         _price = new ChildrensPrice();
         break;
      case NEW_RELEASE:
         _price = new NewReleasePrice();
         break;
      default:
         throw new IllegalArgumentException("Incorrect Price Code");
   }
}
private Price _price;
```

Move getCharge to Price

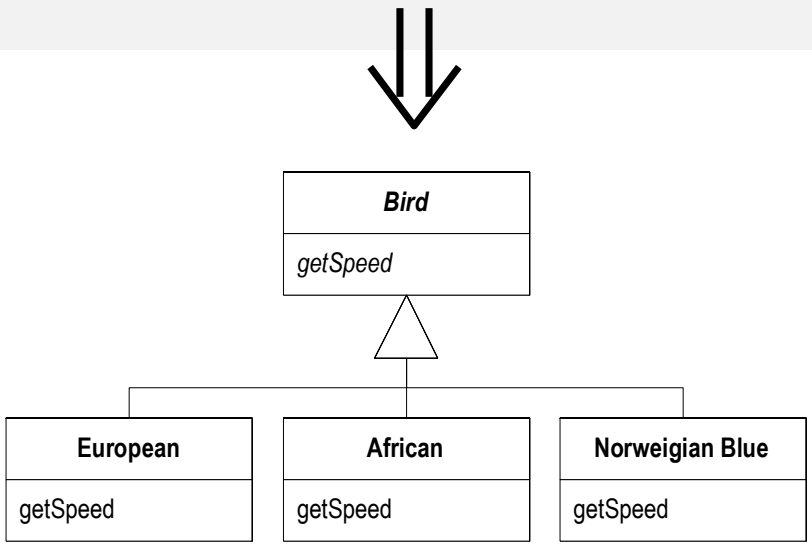
```
class Movie…
double getCharge(int daysRented) {
   return _price.getCharge(daysRented);
}

class Price…
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;
}
```

Replace Conditional With Polymorphism

You have a conditional that chooses different behavior depending on the type of an object.

Move each leg of the conditional to an overriding method in a subclass. Make the original method abstract

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



ThoughtWorks
The art of heavy lifting.

Override getCharge in the subclasses

Do each leg one at a time then…

```
Class Price…
      abstract double getCharge(int daysRented);
```

```
Class RegularPrice…
    double getCharge(int daysRented){
        double result = 2;
        if (daysRented > 2)
            result += (daysRented - 2) * 1.5;
        return result;
    }
Class ChildrensPrice
    double getCharge(int daysRented){
        double result = 1.5;
        if (daysRented > 3)
            result += (daysRented - 3) * 1.5;
        return result;
    }

Class NewReleasePrice…
    double getCharge(int daysRented){
        return daysRented * 3;
    }
```

Do the same with getFrequentRenterPoints()

Similar Statement Methods

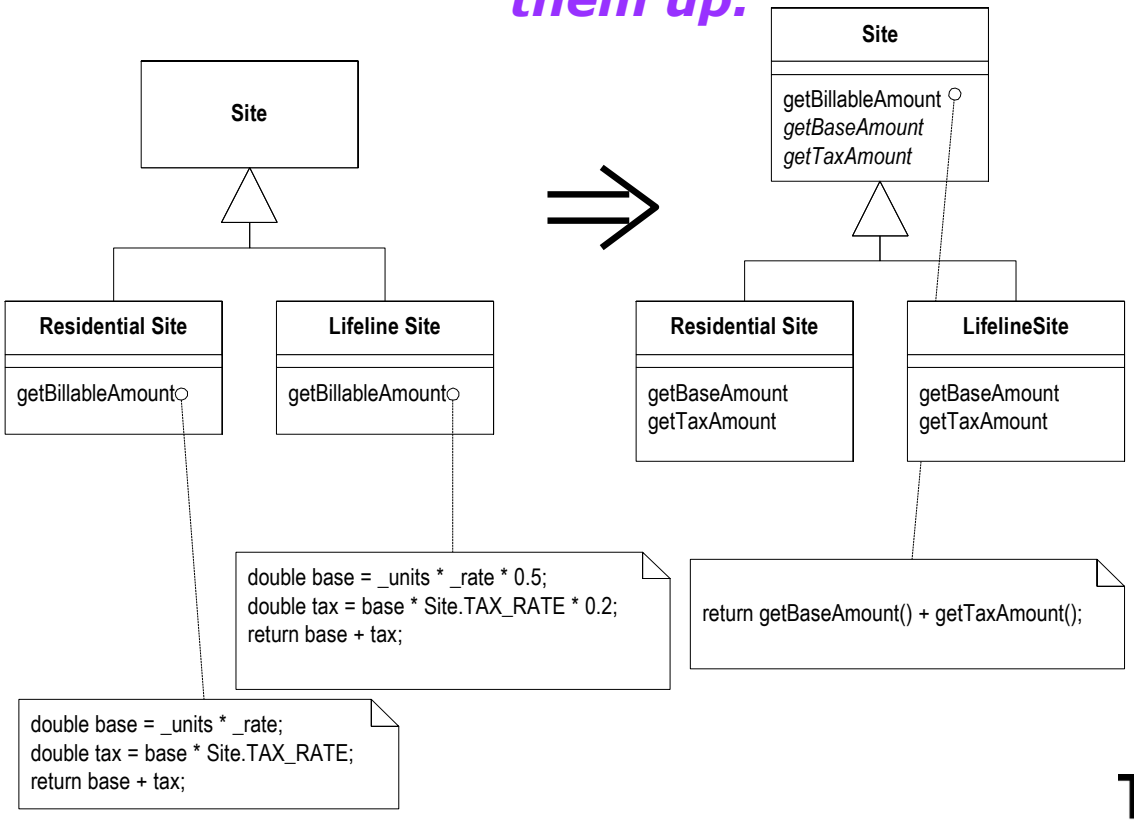
```
 public String statement() {
   Enumeration rentals = _rentals.elements();
   String result = "Rental Record for " + getName() + "\n";
   while (rentals.hasMoreElements()) {
      Rental each = (Rental) rentals.nextElement();
      result += "\t" + each.getMovie().getTitle()+ "\t" +
        String.valueOf(each.getCharge()) + "\n";
         }
   result += "Amount owed is " + String.valueOf(getTotalCharge()) + "\n";
   result += "You earned " + String.valueOf(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();
      result += each.getMovie().getTitle()+ ": " +
      String.valueOf(each.getCharge()) + "<BR>\n";
   }
   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 art of heavy lifting.™

Form Template Method

You have two methods in subclasses that carry out similar steps in the same order, yet the steps are different

Put each step into methods with the same signature, so that the original methods become the same. Then you can pull them up.

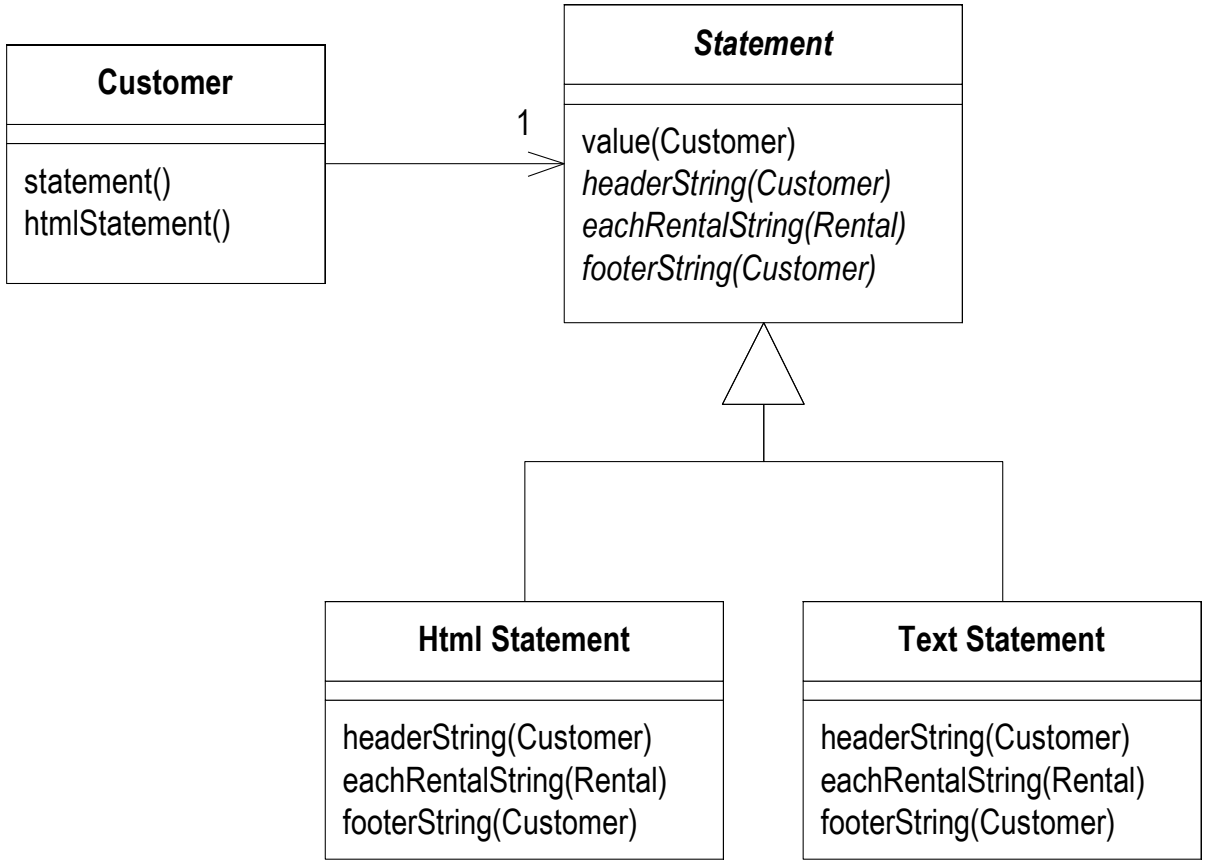


ThoughtWorks®
The art of heavy lifting.™

Create a Statement Strategy with a Template Method

```
class Statement...
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;
    }

abstract String headerString(Customer aCustomer);
abstract String eachRentalString (Rental aRental);
abstract String footerString (Customer aCustomer);
```

Html and Text Strategy Subclasses



In this example

- **We saw a poorly factored program improved**
  - easier to add new statement types
  - easier to add new movie types
- **No debugging during refactoring**
  - appropriate steps reduce chance of bugs
  - small steps make bugs easy to find
- **Illustrated several refactorings**
  - Extract Method
  - Move Method
  - Replace Temp with Query
  - Replace Type Code with State/Strategy
  - Replace Switch with Polymorphism
  - Form Template Method