Refactoring

"Beautiful is better than ugly"

Tim Peters, Zen of Python

Example

```
class Movie:
    CHILDRENS = 2
    REGULAR = 0
    NEW_RELEASE = 1

    _title = None
    _priceCode = None

    def __init__(self, title, priceCode):
        self._title = title
        self._priceCode = priceCode

    def getPriceCode(self):
        return self._priceCode

    def setPriceCode(self, arg):
        self._priceCode = arg

    def getTitle(self):
        return self._title
```

```
class Rental:
    _movie = None
    _daysRented = None

    def __init__(self, movie, daysRented):
        self._movie = movie
        self._daysRented = daysRented

    def getDaysRented(self):
        return self._daysRented

    def getMovie(self):
        return self._movie


class Customer:
    _name = None
    _rentals = []

    def __init__(self, name):
        self._name = name

    def addRental(self, arg):
        self._rentals.append(arg)

    def getName(self):
        return self._name
```

```
class Customer: ...
    def statement(self):
        totalAmount, frequentRenterPoints = (0,0)
        result = "Rental Record for %s\n" % self.getName()

        for each in self._rentals:
            thisAmount = 0
            priceCode = each.getMovie().getPriceCode()
            if priceCode == Movie.REGULAR:
                thisAmount += 2
                if each.getDaysRented() > 2:
                    thisAmount += (each.getDaysRented() - 2) * 1.5
            elif priceCode == Movie.NEW_RELEASE:
                thisAmount += each.getDaysRented() * 3
            elif priceCode == Movie.CHILDRENS:
                thisAmount += 1.5
                if each.getDaysRented() > 3:
                    thisAmount += (each.getDaysRented() - 3) * 1.5

            frequentRenterPoints += 1
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

it works

but...

not readable

long

not object oriented

not extensible

```
class Customer: ...
    def statement(self):
        totalAmount, frequentRenterPoints = (0,0)
        result = "Rental Record for %s\n" % self.getName()

        for each in self._rentals:
            thisAmount = 0
            priceCode = each.getMovie().getPriceCode()
            if priceCode == Movie.REGULAR:
                thisAmount += 2
                if each.getDaysRented() > 2:
                    thisAmount += (each.getDaysRented() - 2) * 1.5
            elif priceCode == Movie.NEW_RELEASE:
                thisAmount += each.getDaysRented() * 3
            elif priceCode == Movie.CHILDRENS:
                thisAmount += 1.5
                if each.getDaysRented() > 3:
                    thisAmount += (each.getDaysRented() - 3) * 1.5

            frequentRenterPoints += 1
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount, frequentRenterPoints = (0,0)
        result = "Rental Record for %s\n" % self.getName()

        for each in self._rentals:
            thisAmount = 0
            priceCode = each.getMovie().getPriceCode()
            if priceCode == Movie.REGULAR:
                thisAmount += 2
                if each.getDaysRented() > 2:
                    thisAmount += (each.getDaysRented() - 2) * 1.5
            elif priceCode == Movie.NEW_RELEASE:
                thisAmount += each.getDaysRented() * 3
            elif priceCode == Movie.CHILDRENS:
                thisAmount += 1.5
                if each.getDaysRented() > 3:
                    thisAmount += (each.getDaysRented() - 3) * 1.5

            frequentRenterPoints += 1
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount, frequentRenterPoints = (0,0)
        result = "Rental Record for %s\n" % self.getName()

        for each in self._rentals:
            thisAmount = 0
            priceCode = each.getMovie().getPriceCode()
            if priceCode == Movie.REGULAR:
                thisAmount += 2
                if each.getDaysRented() > 2:
                    thisAmount += (each.getDaysRented() - 2) * 1.5
            elif priceCode == Movie.NEW_RELEASE:
                thisAmount += each.getDaysRented() * 3
            elif priceCode == Movie.CHILDRENS:
                thisAmount += 1.5
                if each.getDaysRented() > 3:
                    thisAmount += (each.getDaysRented() - 3) * 1.5

            frequentRenterPoints += 1
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount, frequentRenterPoints = (0,0)
        result = "Rental Record for %s\n" % self.getName()

        for each in self._rentals:
            thisAmount = 0
            priceCode = each.getMovie().getPriceCode()
            if priceCode == Movie.REGULAR:
                thisAmount += 2
                if each.getDaysRented() > 2:
                    thisAmount += (each.getDaysRented() - 2) * 1.5
            elif priceCode == Movie.NEW_RELEASE:
                thisAmount += each.getDaysRented() * 3
            elif priceCode == Movie.CHILDRENS:
                thisAmount += 1.5
                if each.getDaysRented() > 3:
                    thisAmount += (each.getDaysRented() - 3) * 1.5

            frequentRenterPoints += 1
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount, frequentRenterPoints = (0,0)
        result = "Rental Record for %s\n" % self.getName()

        for each in self._rentals:
            thisAmount = 0
            priceCode = each.getMovie().getPriceCode()
            if priceCode == Movie.REGULAR:
                thisAmount += 2
                if each.getDaysRented() > 2:
                    thisAmount += (each.getDaysRented() - 2) * 1.5
            elif priceCode == Movie.NEW_RELEASE:
                thisAmount += each.getDaysRented() * 3
            elif priceCode == Movie.CHILDRENS:
                thisAmount += 1.5
                if each.getDaysRented() > 3:
                    thisAmount += (each.getDaysRented() - 3) * 1.5

            frequentRenterPoints += 1
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def amountFor(self, each):
        thisAmount = 0
        priceCode = each.getMovie().getPriceCode()
        if priceCode == Movie.REGULAR:
            thisAmount += 2
            if each.getDaysRented() > 2:
                thisAmount += (each.getDaysRented() - 2) * 1.5
        elif priceCode == Movie.NEW_RELEASE:
            thisAmount += each.getDaysRented() * 3
        elif priceCode == Movie.CHILDRENS:
            thisAmount += 1.5
            if each.getDaysRented() > 3:
                thisAmount += (each.getDaysRented() - 3) * 1.5

        return thisAmount
```

```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = self.amountFor(each)

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

test

:-)

Refactoring

?

cleaning up code

Refactoring

=

cleaning up code

small changes

test

small changes

test

small changes

test

- **Composing Methods**
  - Extract Method
  - Inline Method
  - Inline Temp
  - Replace Temp with Query
  - Introduce Explaining Variable
  - Split Temporary Variable
  - Remove Assignments to Parameters
  - Replace Method with Method Object
  - Substitute Algorithm
- **Moving Features Between Objects**
  - Move Method
  - Move Field
  - Extract Class
  - Inline Class
  - Hide Delegate
  - Remove Middle Man
  - Introduce Foreign Method
  - Introduce Local Extension
- **Simplifying Conditional Expressions**
  - Decompose Conditional
  - Consolidate Conditional Expression
  - Consolidate Duplicate Conditional Fragments
  - Remove Control Flag
  - Replace Nested Conditional with Guard Clauses
  - Replace Conditional with Polymorphism
  - Introduce Null Object
  - Introduce Assertion
- **Organizing Data**
  - Self Encapsulate Field
  - Replace Data Value with Object
  - Change Value to Reference
  - Change Reference to Value
  - Replace Array with Object
  - Duplicate Observed Data
  - Change Unidirectional Association to Bidirectional
  - Change Bidirectional Association to Unidirectional
  - Replace Magic Number with Symbolic Constant
  - Encapsulate Field
  - Encapsulate Collection
  - Replace Record with Data Class
  - Replace Type Code with Class
  - Replace Type Code with Subclasses
  - Replace Type Code with State/Strategy
  - Replace Subclass with Fields

- **Dealing with Generalization**
  - Pull Up Field
  - Pull Up Method
  - Pull Up Constructor Body
  - Push Down Method
  - Push Down Field
  - Extract Subclass
  - Extract Superclass
  - Extract Interface
  - Collapse Hierarchy
  - Form Template Method
  - Replace Inheritance with Delegation
  - Replace Delegation with Inheritance
- **Making Method Calls Simpler**
  - Rename Method
  - Add Parameter
  - Remove Parameter
  - Separate Query from Modifier
  - Parameterize Method
  - Replace Parameter with Explicit Methods
  - Preserve Whole Object
  - Replace Parameter with Method
  - Introduce Parameter Object
  - Remove Setting Method
  - Hide Method
  - Replace Constructor with Factory Method
  - Encapsulate Downcast
  - Replace Error Code with Exception
  - Replace Exception with Test

- **Big Refactorings**
  - Tease Apart Inheritance
  - Convert Procedural Design to Objects
  - Separate Domain from Presentation
  - Extract Hierarchy

- **Composing Methods**
  - Extract Method
  - Inline Method
  - Inline Temp
  - Replace Temp with Query
  - Introduce Explaining Variable
  - Split Temporary Variable
  - Remove Assignments to Parameters
  - Replace Method with Method Object
  - Substitute Algorithm
- **Moving Features Between Objects**
  - Move Method
  - Move Field
  - Extract Class
  - Inline Class
  - Hide Delegate
  - Remove Middle Man
  - Introduce Foreign Method
  - Introduce Local Extension
- **Simplifying Conditional Expressions**
  - Decompose Conditional
  - Consolidate Conditional Expression
  - Consolidate Duplicate Conditional Fragments
  - Remove Control Flag
  - Replace Nested Conditional with Guard Clauses
  - Replace Conditional with Polymorphism
  - Introduce Null Object
  - Introduce Assertion

- **Organizing Data**
  - Self Encapsulate Field
  - Replace Data Value with Object
  - Change Value to Reference
  - Change Reference to Value
  - Replace Array with Object
  - Duplicate Observed Data
  - Change Unidirectional Association to Bidirectional
  - Change Bidirectional Association to Unidirectional
  - Replace Magic Number with Symbolic Constant
  - Encapsulate Field
  - Encapsulate Collection
  - Replace Record with Data Class
  - Replace Type Code with Class
  - Replace Type Code with Subclasses
  - Replace Type Code with State/Strategy
  - Replace Subclass with Fields

- **Composing Methods**
  - Extract Method
  - Inline Method
  - Inline Temp
  - Replace Temp with Query
  - Introduce Explaining Variable
  - Split Temporary Variable
  - Remove Assignments to Parameters
  - Replace Method with Method Object
  - Substitute Algorithm
- **Moving Features Between Objects**
  - Move Method
  - Move Field
  - Extract Class
  - Inline Class
  - Hide Delegate
  - Remove Middle Man
  - Introduce Foreign Method
  - Introduce Local Extension
- **Simplifying Conditional Expressions**
  - Decompose Conditional
  - Consolidate Conditional Expression
  - Consolidate Duplicate Conditional Fragments
  - Remove Control Flag
  - Replace Nested Conditional with Guard Clauses
  - Replace Conditional with Polymorphism
  - Introduce Null Object
  - Introduce Assertion
- **Organizing Data**
  - Self Encapsulate Field
  - Replace Data Value with Object
  - Change Value to Reference
  - Change Reference to Value
  - Replace Array with Object
  - Duplicate Observed Data
  - Change Unidirectional Association to Bidirectional
  - Change Bidirectional Association to Unidirectional
  - Replace Magic Number with Symbolic Constant
  - Encapsulate Field
  - Encapsulate Collection
  - Replace Record with Data Class
  - Replace Type Code with Class
  - Replace Type Code with Subclasses
  - Replace Type Code with State/Strategy
  - Replace Subclass with Fields

Rename Variables

```
class Customer: ...
    def amountFor(self, each):
        thisAmount = 0
        priceCode = each.getMovie().getPriceCode()
        if priceCode == Movie.REGULAR:
            thisAmount += 2
            if each.getDaysRented() > 2:
                thisAmount += (each.getDaysRented() - 2) * 1.5
        elif priceCode == Movie.NEW_RELEASE:
            thisAmount += each.getDaysRented() * 3
        elif priceCode == Movie.CHILDRENS:
            thisAmount += 1.5
            if each.getDaysRented() > 3:
                thisAmount += (each.getDaysRented() - 3) * 1.5

        return thisAmount
```

```
class Customer: ...
    def amountFor(self, each):
        thisAmount = 0
        priceCode = each.getMovie().getPriceCode()
        if priceCode == Movie.REGULAR:
            thisAmount += 2
            if each.getDaysRented() > 2:
                thisAmount += (each.getDaysRented() - 2) * 1.5
        elif priceCode == Movie.NEW_RELEASE:
            thisAmount += each.getDaysRented() * 3
        elif priceCode == Movie.CHILDRENS:
            thisAmount += 1.5
            if each.getDaysRented() > 3:
                thisAmount += (each.getDaysRented() - 3) * 1.5

        return thisAmount
```

```
class Customer: ...
    def amountFor(self, aRental):
        result = 0
        priceCode = aRental.getMovie().getPriceCode()
        if priceCode == Movie.REGULAR:
            result += 2
            if aRental.getDaysRented() > 2:
                result += (aRental.getDaysRented() - 2) * 1.5
        elif priceCode == Movie.NEW_RELEASE:
            result += aRental.getDaysRented() * 3
        elif priceCode == Movie.CHILDRENS:
            result += 1.5
            if aRental.getDaysRented() > 3:
                result += (aRental.getDaysRented() - 3) * 1.5

        return result
```

test

:-)

trivial?

"Readability counts."

Tim Peters, Zen of Python

Move Method

```
class Customer: ...
    def amountFor(self, aRental):
        result = 0
        priceCode = aRental.getMovie().getPriceCode()
        if priceCode == Movie.REGULAR:
            result += 2
            if aRental.getDaysRented() > 2:
                result += (aRental.getDaysRented() - 2) * 1.5
        elif priceCode == Movie.NEW_RELEASE:
            result += aRental.getDaysRented() * 3
        elif priceCode == Movie.CHILDRENS:
            result += 1.5
            if aRental.getDaysRented() > 3:
                result += (aRental.getDaysRented() - 3) * 1.5

        return result
```

```
class Customer: ...
    def amountFor(self, aRental):
        result = 0
        priceCode = aRental.getMovie().getPriceCode()
        if priceCode == Movie.REGULAR:
            result += 2
            if aRental.getDaysRented() > 2:
                result += (aRental.getDaysRented() - 2) * 1.5
        elif priceCode == Movie.NEW_RELEASE:
            result += aRental.getDaysRented() * 3
        elif priceCode == Movie.CHILDRENS:
            result += 1.5
            if aRental.getDaysRented() > 3:
                result += (aRental.getDaysRented() - 3) * 1.5

        return result
```

```
class Rental: ...
    def getCharge(self):
        result = 0
        priceCode = self.getMovie().getPriceCode()
        if priceCode == Movie.REGULAR:
            result += 2
            if self.getDaysRented() > 2:
                result += (self.getDaysRented() - 2) * 1.5
        elif priceCode == Movie.NEW_RELEASE:
            result += self.getDaysRented() * 3
        elif priceCode == Movie.CHILDRENS:
            result += 1.5
            if self.getDaysRented() > 3:
                result += (self.getDaysRented() - 3) * 1.5
        return result


class Customer: ...
    def amountFor(self, aRental):
        return aRental.getCharge()
```

test

:-)

```
class Customer: ...
     def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = self.amountFor(each)

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = self.amountFor(each)

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = each.getCharge()

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def amountFor(self, aRental):
        return aRental.getCharge()
```

test

:-)

What I should refactor?

"If it stinks, change it!"

Kent Becks Grandma

Bad Smells in Code

- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- Switch Statements
- Parallel Inheritance Hierarchies

- Lazy Class
- Speculative Generality
- Temporary Field
- Message Chains
- Middle Man
- Inappropriate Intimacy
- Alternative Classes with Different Interfaces
- Incomplete Library Class
- Data Class
- Refused Bequest
- Comments

- **Duplicated Code**
- **Long Method**
- **Large Class**
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- **Switch Statements**
- Parallel Inheritance Hierarchies
- Lazy Class
- Speculative Generality
- Temporary Field
- Message Chains
- Middle Man
- Inappropriate Intimacy
- Alternative Classes with Different Interfaces
- Incomplete Library Class
- Data Class
- Refused Bequest
- Comments

- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- Switch Statements
- Parallel Inheritance Hierarchies

- Lazy Class
- Speculative Generality
- Temporary Field
- Message Chains
- Middle Man
- Inappropriate Intimacy
- Alternative Classes with Different Interfaces
- Incomplete Library Class
- Data Class
- Refused Bequest
- **Comments**

?

```
# create order object
order.create(...)
```

Replace Temp with Query

```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = each.getCharge()

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

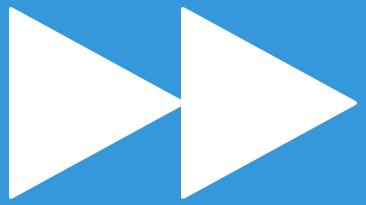
```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = each.getCharge()

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = each.getCharge()

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), each.getCharge())
            totalAmount += each.getCharge()

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

test

:-)

Performance?



```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = each.getCharge()

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = each.getCharge()

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

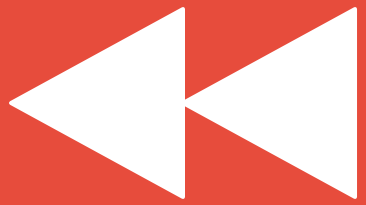
```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = each.getCharge()

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Rental: ...
     def getFrequentPoints(self):
        if (self.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (self.getDaysRented() > 1):
            return 2
        else:
            return 1


class Customer: ...
    def getTotalCharge(self):
        result = 0

        for each in self._rentals:
            result += each.getCharge()

        return result

    def getTotalFrequentRenterPoints(self):
        result=0

        for each in self._rentals:
            result += each.getFrequentPoints()

        return result
```

```
class Customer: ...
    def statement(self):
        totalAmount = 0
        frequentRenterPoints = 0
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:

            thisAmount = each.getCharge()

            # add frequent renter points
            frequentRenterPoints += 1

            # add bonus for a two day new release rental
            if (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) & (each.getDaysRented() > 1):
                frequentRenterPoints += 1

            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), thisAmount)
            totalAmount += thisAmount

        # add footer lines
        result += "Amount owed is %s\n" % totalAmount
        result += "You earned %s frequent renter points" % frequentRenterPoints

        return result
```

```
class Customer: ...
    def statement(self):
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:
            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), each.getCharge())

        # add footer lines
        result += "Amount owed is %s\n" % self.getTotalCharge()
        result += "You earned %s frequent renter points" % self.getTotalFrequentRenterPoints()

        return result
```

```
class Customer: ...
    def htmlStatement(self):
        result = "<h1>Rentals for <em>%s</em></h1><p>\n" % self.getName()

        for each in self._rentals:
            # show figures for this rental
            result += "%s: %s</br>\n" % (each.getMovie().getTitle(), each.getCharge())

        # add footer lines
        result += "<p>You owe <em>%s</em></p>\n" % self.getTotalCharge()
        result += "On this rental you earned <em>%s</em> frequent renter points</p>"
                    % self.getTotalFrequentRenterPoints()

        return result
```



```
class Customer: ...
    def getTotalCharge(self):
        result = 0

        for each in self._rentals:
            result += each.getCharge()

        return result

    def getTotalFrequentRenterPoints(self):
        result=0

        for each in self._rentals:
            result += each.getFrequentPoints()

        return result

    def statement(self):
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:
            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), each.getCharge())

        # add footer lines
        result += "Amount owed is %s\n" % self.getTotalCharge()
        result += "You earned %s frequent renter points" % self.getTotalFrequentRenterPoints()

        return result
```

```
class Customer: ...
    def getTotalCharge(self):
        result = 0

        for each in self._rentals:
            result += each.getCharge()

        return result

    def getTotalFrequentRenterPoints(self):
        result=0

        for each in self._rentals:
            result += each.getFrequentPoints()

        return result

    def statement(self):
        result = "Rental Record for %s\n" % self.getName()

        # determine amounts for each line
        for each in self._rentals:
            # show figures for this rental
            result += "\t%s\t%s\n" % (each.getMovie().getTitle(), each.getCharge())

        # add footer lines
        result += "Amount owed is %s\n" % self.getTotalCharge()
        result += "You earned %s frequent renter points" % self.getTotalFrequentRenterPoints()

        return result
```

We Loop 3 Times!

PERFORMANCE???

```
class Customer: ...
    def getTotalCharge(self):
        result = 0

        for each in self._rentals:
            result += each.getCharge()

        return result

    def getTotalFrequentRenterPoints(self):
        result=0

        for each in self._rentals:
            result += each.getFrequentPoints()

        return result

    def statement(self):
        result = "Rental Record for %s\n" % self.getName()

        for each in self._rentals:
            # show figures for this rental
            result += "\t%s\t%s. (%s of %s)\n" % (each.getMovie().getTitle(), each.getCharge(),
                                                  each.getCharge()/each.getTotalCharge()*100,
                                                  each.getTotalCharge())
        # add footer lines
        result += "Amount owed is %s\n" % self.getTotalCharge()
        result += "You earned %s frequent renter points" % self.getTotalFrequentRenterPoints()

        return result
```

```
class Customer: ...
    def getTotalCharge(self):
        result = 0

        for each in self._rentals:
            result += each.getCharge()

        return result

    def getTotalFrequentRenterPoints(self):
        result=0

        for each in self._rentals:
            result += each.getFrequentPoints()

        return result

    def statement(self):
        result = "Rental Record for %s\n" % self.getName()

        for each in self._rentals:
            # show figures for this rental
            result += "\t%s\t%s. (%s of %s)\n" % (each.getMovie().getTitle(), each.getCharge(),
                                                  each.getCharge()/each.getTotalCharge()*100,
                                                  each.getTotalCharge())
        # add footer lines
        result += "Amount owed is %s\n" % self.getTotalCharge()
        result += "You earned %s frequent renter points" % self.getTotalFrequentRenterPoints()

        return result
```

$$T(n)=n^2$$

Databases?

SELECT with 3 JOINS n times

Temp Vars

?

Bad Smell

- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- Switch Statements
- Parallel Inheritance Hierarchies

- Lazy Class
- Speculative Generality
- Temporary Field
- Message Chains
- Middle Man
- Inappropriate Intimacy
- Alternative Classes with Different Interfaces
- Incomplete Library Class
- Data Class
- Refused Bequest
- Comments

Temp Vars

≠

Bad Smell

Problems of Refactoring

Changing Database Structure

Changing Interfaces

When I should refactor?

The Rule of Three

Add Function

Fixing a Bug

Code Review

Tools

References

- Fowler, Martin Beck, Kent: "Refactoring / improving the design of existing code" 1999
- Kerievsky Joshua: "Refactoring to patterns" 2005

Questions?