## 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());

}

http://www.refactoring.com/catalog/arrow.gif

void printOwing() {

printBanner();

printDetails(getOutstanding());

}

void printDetails (double outstanding) {

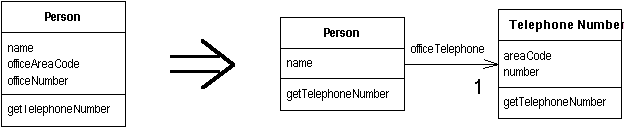
System.out.println ("name: " + \_name);

System.out.println ("amount " + outstanding);

}

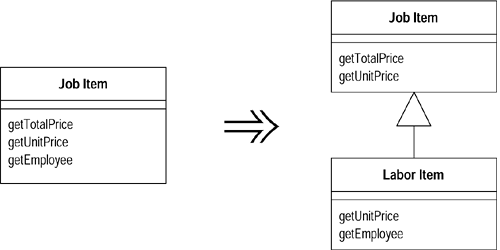
## Extract Class

You have one class doing work that should be done by two.  
**Create a new class and move the relevant fields and methods from the old class into the new class.**



## Extract Sub Class

A class has features that are used only in some instances.  
**Create a subclass for that subset of features.**



## Introduce Explaining Variable

You have a complicated expression.  
**Put the result of the expression, or parts of the expression, in a temporary variable with a name that explains the purpose.**

if ((platform.toUpperCase().indexOf("MAC") > -1) &&

(browser.toUpperCase().indexOf("IE") > -1) &&

wasInitialized() && resize > 0 ) {

// do something

}

graphics/arrow.gif

final boolean isMacOs = platform.toUpperCase().indexOf("MAC") > -1;

final boolean isIEBrowser = browser.toUpperCase().indexOf("IE") > -1;

final boolean wasResized = resize > 0;

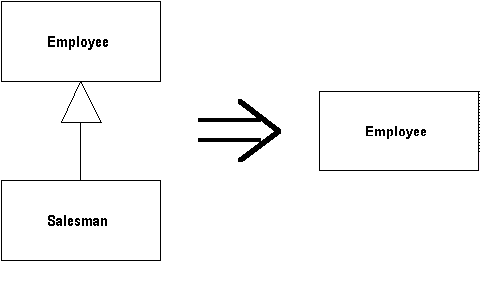
if (isMacOs && isIEBrowser && wasInitialized() && wasResized) {

// do something

}

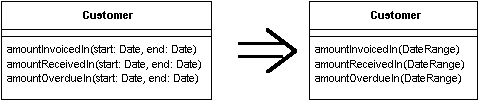
## Collapse Hierarchy

A superclass and subclass are not very different.  
**Merge them together.**



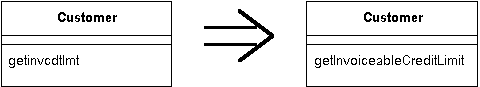
## Introduce Parameter Object

You have a group of parameters that naturally go together.  
**Replace them with an object.**



## Rename

The name of a method does not reveal its purpose.  
**Change the name of the method.**



## 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 NORWEGIAN\_BLUE:

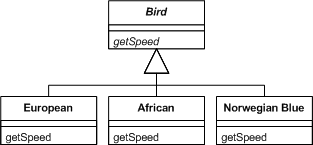
return (\_isNailed) ? 0 : getBaseSpeed(\_voltage);

}

throw new RuntimeException ("Should be unreachable");

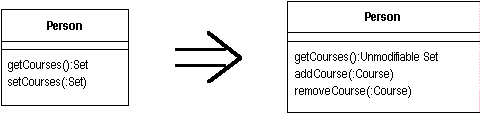
}

http://www.refactoring.com/catalog/arrow.gif



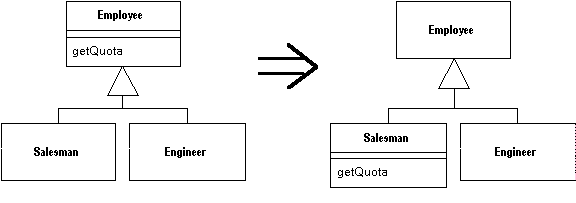
## Encapsulate Collection

A method returns a collection.  
**Make it return a read-only view and provide add/remove methods.**



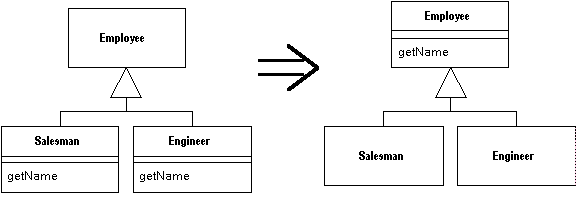
## Push Down Field/Method

Behavior on a superclass is relevant only for some of its subclasses.  
**Move it to those subclasses.**



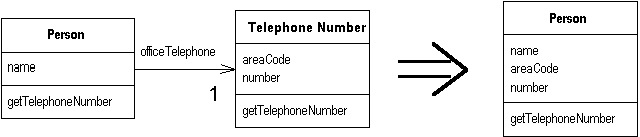
## Pull Up Field/Method

You have methods with identical results on subclasses.  
**Move them to the superclass.**



## Inline Class

A class isn't doing very much.  
**Move all its features into another class and delete it.**



## Inline Method

A method’s body is just as clear as its name.  
**Put the method’s body into the body of its callers and remove the method.**

int getRating() {

return (moreThanFiveLateDeliveries()) ? 2 : 1;

}

boolean moreThanFiveLateDeliveries() {

return \_numberOfLateDeliveries > 5;

}

graphics/arrow.gif

int getRating() {

return (\_numberOfLateDeliveries > 5) ? 2 : 1;

}

## Inline Temp

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

double basePrice = anOrder.basePrice();

return (basePrice > 1000)

graphics/arrow.gif

return (anOrder.basePrice() > 1000)

## Move Method

A method is, or will be, using or used by more features of another class than the class on which it is defined.  
**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.**

